Universidad de Cádiz

Escuela Superior de Ingeniería
Dirección

C/ Chile, 1. 11002 Cádiz
Tlf. 956015100
Fax 956015101

Escuela Superior de Ingeniería

PROYECTO FIN DE CARRERA

Simulador de Fútbol Online desarrollado bajo Ruby On Rails

Simulador de Fútbol Online desarrollado bajo Ruby On Rails

Carlos Rafael Belizón Ibáñez,
Juan Manuel Dodero Beardo[1]
Manuel Palomo Duarte[2]

Avda. Almirante León Herrero, 29 1º C, San Fernando, Cádiz, CP/11100,
652 019 117, carlos.belizon.ibanez@gmail.com

[1][2]Aulario Simón Bolívar (Cádiz)
Duque de Nájera, 16
CP/11002, Cádiz

En este proyecto desarrollaremos un sistema web de gestión de clubes de fútbol orientado al disfrute del usuario como de si un videojuego se tratara. Este sistema nos ha permitido conocer los entresijos del desarrollo software web mediante el uso del framework de desarrollo rápido de aplicaciones web Ruby On Rails.

Palabras Clave: Ruby, Rails, Fútbol, Gestión, Simulador

1. Introducción y objetivos del proyecto

Este proyecto se inició para completar la formación académica de la titulación sede Ingeniería Técnica en Informática de Gestión de la Escuela Superior de Ingeniería de Cádiz de la Universidad de Cádiz.

Gracias a ella, pudimos aprender cómo desarrollar un producto totalmente funcional usando para ello el uso de un framework basado en el patrón Modelo-Vista-Controlador Ruby On Rails.

Además, comprendimos la dificultad que conlleva el desarrollo de un sistema software, desde la misma etapa de conceptualización del sistema.

El objetivo del proyecto es proporcionar un videojuego web basado en la gestión de clubes de fútbol que se adapte a las condiciones de licencia del software libre que sea capaz de competir en un futuro con las alternativas actualmente existentes en el mercado actual.

Al concluir el desarrollo, la aplicación ha de ser capaz de desempeñar todas las funcionalidades descritas en la etapa de Análisis

2. Análisis del sistema

Para conocer los requisitos funcionales del sistema a desarrollar nos reunimos varias veces con nuestro tutor de proyecto en las primeras semanas de desarrollo del sistema.

De dichas reuniones, fuimos capaces de llegar a numerosas conclusiones sobre cuáles serían los límites y funcionalidades que habríamos de desarrollar en el proyecto, para poder crear la Especificación de Requerimentos (SRS) [1].

2.1. Características de los usuarios

El producto está dirigido a todo usuario que sepa manejar un navegador web y que esté interesado en la simulación de la gestión de un club de fútbol, así como conocer las reglas y premisas básicas de negocio del deporte. Identificamos dos roles bien diferenciados en nuestro sistema:

1. **Usuario Jugador:** Se refiere a una persona que se registra en el portal web y que participará en la gestión de un club al que se le asignará. Este usuario podrá conectarse al sistema e interactuar con el mismo a través de la interfaz web.
2. **Usuario Administrador:** Es un tipo de usuario que será capaz de modificar el comportamiento del sistema a través del menú de administración de nuestra aplicación para llevar a cabo las acciones de gestión de la competición de nuestro sistema.

2*.2. Requisitos de rendimiento*

Este aspecto es fundamental en el análisis del sistema, se ha de tener en cuenta que la aplicación es de tipo servidor y que a ella estarán conectados concurrentemente un número de usuarios elevado.

Se ha considerado también el rendimiento que nos ofrece el diseño web realizado. Para que nuestra página web sea compatible con la mayoría de dispositivos y se ejecute lo suficientemente fluida hemos decidido no usar la tecnología propietaria Flash, usando para la renderización de efectos la librería JQuery en combinación con hojas de estilo CSS.

El rendimiento de la aplicación podrá ser menor en la primera instanciación de la aplicación debido a que no se habrá cacheado ningún contenido.

2.3. Requisitos de interfaces externas

Aquí detallaremos los requisitos de conexión a otros sistemas hardware o software con los que vamos a interactuar.

2.3.1. Interfaces de usuario

Podemos dividir los requisitos de interfaz de usuario según tres roles:

- **Usuario Jugador y Usuario Administrador:**
    - Interfaz atractiva para los usuarios: Es fundamental para la buena aceptación de nuestra aplicación el crear una interfaz lo suficientemente clara, dinámica y estética para que el usuario sienta una inmersión en la experiencia de juego. Una interfaz no atractiva sólo traerá problemas de rechazo y disconformidad por parte del usuario.

      Para que la interfaz sea sencilla de modificar y tratándose de una aplicación web usaremos hojas de estilo CSS, así todo el apartado de diseño gráfico podrá ser modificado posteriormente sin que por ello afecte al contenido ni la estructura del documento HTML.
    - Una característica de nuestra aplicación es que el sistema de avisos está embebido dentro de la página web y es muy parecido al sistema de

notificación de los sistemas operativos Mac OS X y de la distribución Linux Ubuntu.

- **Superusuario:** En este caso la interfaz corresponderá a la que tenga el propio sistema operativo donde esté instalado la aplicación.

2.3.2. Interfaz con el hardware

Las copias de seguridad del sistema se realizarán de forma periódica cada día, semana y mes de forma incremental y utilizando la herramienta rsync. Dichas copias de seguridad se almacenarán en otro servidor dedicado exclusivamente al almacenaje de las copias.

2.3.3. Interfaz con el software

La aplicación web correrá en el lado del servidor en un sistema Linux con Ubuntu Server 12.04. Mientras que en el lado cliente podrá ejecutarse en cualquier sistema que disponga de un navegador web que cumpla los estándares 2.0 w3c.

2.4. Requisitos de Rendimiento

Este aspecto es fundamental en el análisis del sistema, se ha de tener en cuenta que la aplicación es de tipo servidor y que a ella estarán conectados concurrentemente un número de usuarios elevado.

Se ha considerado también el rendimiento que nos ofrece el diseño web realizado. Para que nuestra página web sea compatible con la mayoría de dispositivos y se ejecute lo suficientemente fluida hemos decidido no usar la tecnología propietaria Flash, usando para la renderización de efectos la librería JQuery en combinación con hojas de estilo CSS.

El rendimiento de la aplicación podrá ser menor en la primera instanciación de la aplicación debido a que no se habrá cacheado ningún contenido.

2.5. Requisitos de Información

El sistema almacenará en forma de tablas la siguiente información relativa a las siguientes entidades:

User Se almacenan los datos de cada uno de los usuarios que interactúan con la aplicación.

Club Se almacenan los datos de los clubes que participan en las ligas.

League Se almacenan los datos de las ligas que existen en el sistema.

Season Se almacenan los datos de las temporadas correspondientes a cada liga del sistema.

Round Se almacenan los datos de las jornadas de cada temporada.

MatchGeneral Se almacenan los partidos que se han de jugar cada jornada.

MatchDetail Se almacenan las acciones que tienen lugar en cada partido.

LineUp Se almacenan las alineaciones de cada equipo y partido.

ClubFinancesRound Se almacenan los datos de finanzas correspondientes a cada jornada y club.

Player Se almacenan los datos de los jugadores de fútbol de los clubes.

Offer Se almacenan los datos de las ofertas que se realizan los usuarios de la aplicación sobre los jugadores de fútbol.

Training Se almacenan los datos de los entrenamientos de los jugadores de fútbol.

AdminMessages Se almacenan los mensajes de administración hacia los usuarios.

3. Diseño del sistema

Para el diseño del sistema tuvimos que tomar algunas decisiones básicas, sobre todo orientadas a qué tecnologías emplear.

Entre ellas destacamos el uso de la tecnología de Software Libre Ruby On Rails, basado en el lenguaje de script Ruby. Dicho framework nos proporciona todas las herramientas necesarias para el desarrollo del sistema incluyendo el de la interfaz cliente.

3.1. Arquitectura

La arquitectura que emplearemos es la que mejor se ajusta a las aplicaciones cliente/servidor, aplicando el patrón Modelo-Vista-Controlador que nos proporciona de base Rails [2].

El patrón Modelo-Vista-Controlador divide el software en tres capas bien diferenciadas que se adaptan muy bien a la tecnología Cliente-Servidor. Dichas capas son:

- **Modelo:** Esta capa es la relativa a la representación de la información con la que el proyecto interactuará. Está compuesto por las clases necesarias para el acceso, modificación, búsqueda y demás operaciones con los datos.
- **Vista:** En esta capa agruparemos todas las clases necesarias para la representación de la interfaz con la que nuestros usuarios interactuarán.
- **Controlador:** Será la capa que nos permita gestionar las peticiones que lleguen a nuestro servidor.

Los modelos usan el motor de persistencia ActiveRecord integrado en Rails y basado en el patrón de mismo nombre Active Record [3]. Esta clase es la que en última instancia ejecuta las órdenes de escritura y lectura en lenguaje SQL en el SGBD que el administrador haya decidido usar, en nuestro caso SQLite3.

La vista son los ficheros con extensión .erb basados en el sistema de plantillas eRuby.

3.2. Diagrama de Componentes

Podemos ver el diagrama de componentes en la Figura 1.

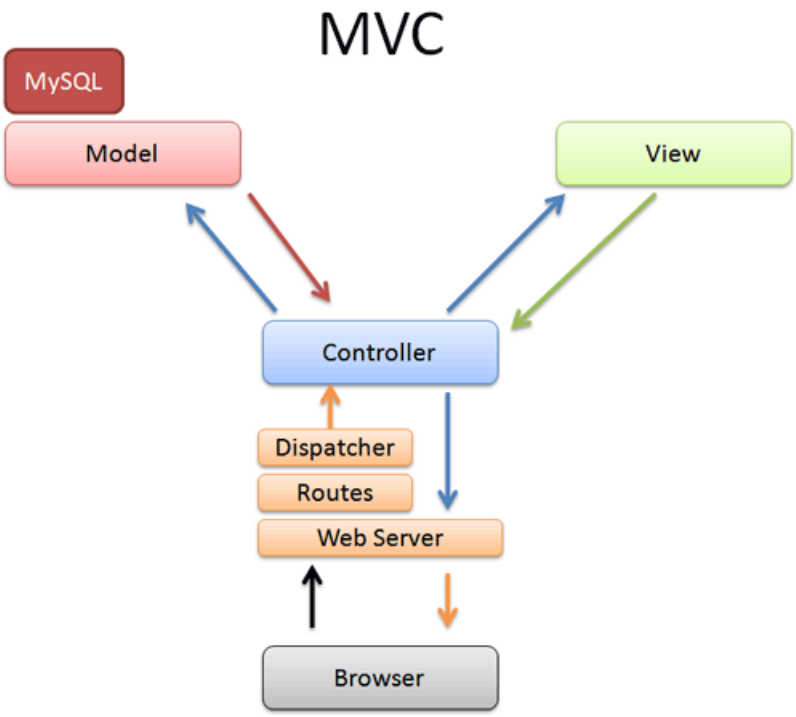

Figura 1. Diagrama de componentes

El framework Ruby on Rails hace especial hincapié en tres conceptos muy básicos que caracterizan toda su estructura, que son:

- **Convention over Configuration (COC):** Convención frente a configuración.
- **Don't Repeat Yourself (DRY):** No te repitas
- **Keep It Simple (KIS):** Mantenlo simple.

Gracias a ello seguiremos la estructura de directorios en árbol para nuestro código fuente que genera Rails para una aplicación vacía.

3.3. Entorno de ejecución

En la Figura 2 podemos ver el diagrama de despliegue de la aplicación puesta en marcha en cualquiera de los entornos, ya sea este de desarrollo, test o producción.

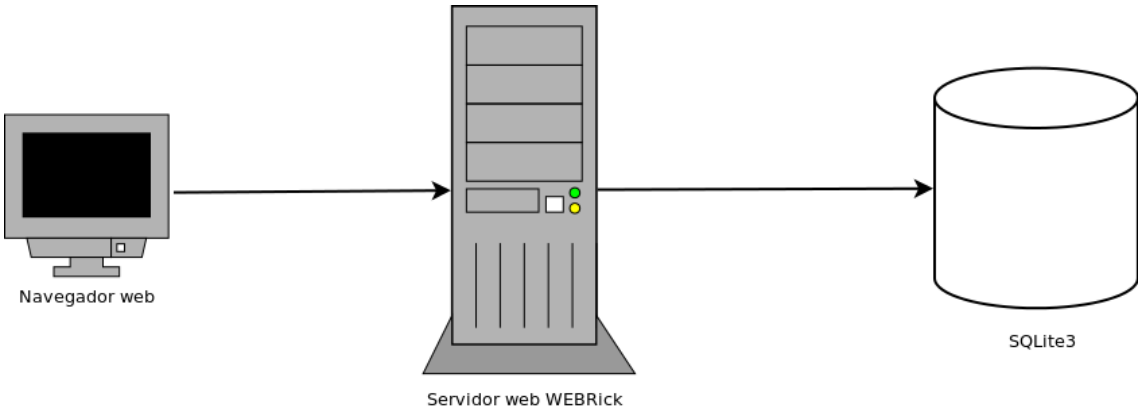

Figura 2. *Diagrama de estructura*

El sistema (servidor) interactuará con uno o más clientes web (navegadores). El servidor, a su vez estará formado por un servidor de aplicaciones web, en nuestro caso Webrick, y el sistema de gestión de base de datos será SQLite3.

Gracias a esta sencilla configuración no tendremos que configurar ninguna cuenta ni servicio en nuestro servidor web.

3.4. Diagrama de clases conceptuales

A continuación mostramos algunos de los modelos de clases conceptuales en notación UML [4].

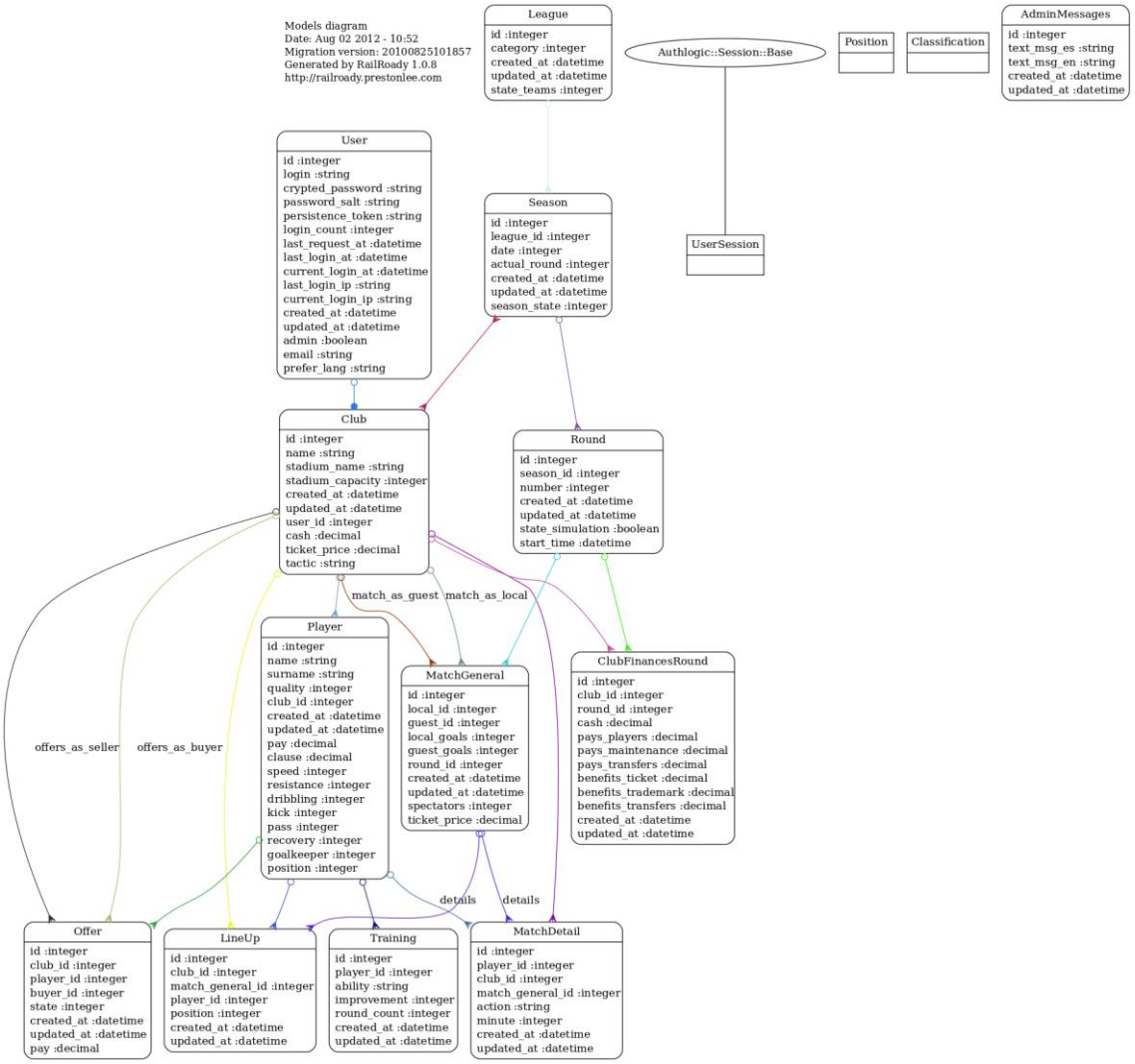

Figura 3. Diagrama de Clases UML de la capa modelo.

Figura 4. Diagrama de Clases UML de la capa controlador.

3.5. Sistema de Simulación

Este apartado de la memoria requiere una especial mención; ya que es el sistema que hará que el que los cambios y modificaciones a la táctica, alineación, entrenamientos y precio de entradas que realice en su equipo, tengan un reflejo directo en el desarrollo de los partidos que conforman la temporada que disputa.

Para perfilar el sistema de simulación de nuestro proyecto de fin de carrera tuvimos varias reuniones con nuestros tutores y acotamos la forma de simular un partido en base a la táctica y alineación que presentan los clubes que disputarán el partido.

4. Implementación

Para la implementación del software hemos utilizado los siguientes lenguajes:

- Ruby como lenguaje servidor.
- HTML como lenguaje de marcado.
- CoffeScript como metalenguaje para el desarrollo en JavaScript[5].
- SASS como metalenguaje para el desarrollo de hojas de estilo [6].
- Y las aplicaciones utilizadas para codificar a de una forma sencilla y productiva:
- El editor de texto extensible SublimeText2 [7].
- El depurador de Firefox [8] Firebug. [9].
- El controlador de versiones GIT [10].
- El framework CSS TwitterBootstrap. [11].

5. Pruebas y validación

En la fase de análisis comprobamos la validez de nuestros requisitos identificando de forma correcta cuáles serían las funcionalidades, restricciones y el ámbito de desarrollo

y ejecución de nuestra aplicación; además de comprobar que dichos requisitos no se contradicen ni acoplan.

Posteriormente validamos que la especificación de requisitos era correcta con nuestro tutor de proyecto de fin de carrera.

Las pruebas efectuadas sobre el entorno de desarrollo han sido de tipo manual en todo momento debido a la necesidad de cumplir con el calendario de entrega de la aplicación.

6. Conclusiones y trabajo futuro

Este proyecto ha servido para poder completar la formación académica recibida en la Escuela Superior de Ingeniería orientada a la finalización de la titulación de Ingeniería Técnica en Informática de Gestión de la Universidad de Cádiz.

El desarrollo de este sistema software nos ha permitido conocer en profundidad cómo se realiza el desarrollo de una aplicación a través de la utilización de un framework y las dificultades y retos que esto conlleva.

Este sistema es altamente ampliable debido a la naturaleza de videojuego del mismo. Una de las características a mejorar y ampliar sería la de la creación de un sistema de simulación de partidos basadas en tecnologías avanzadas de inteligencia artificial, lo cual llevaría asociada una plena experiencia de usuario.

7. Referencias

1. Raymond Turner. The foundations of specification, (2004).

2. Dave Thomas y David Heinemeier Hansson Sam Ruby. Agile Web Development with Rails, 3ed Edition. The Pragmatic Programmers, (2010).

3. Wikipedia. Active Record Pattern.

http://en.wikipedia.org/wiki/Active_record_pattern

4. Kendall Scott Martin Fowler. UML Distilled: A Brief Guide to the Standard Object Modeling Language (3erd Edition). Addison-Wesley Professional.

5. Jeremy Ashkenas. CoffeScript. http://www.coffeescript.org

6. Hampton Catlin. SASS. http://www.sass-lang.com

7. Jon Skinner. Página oficial de SublimeText2.

8. Mozilla Foundation. Página web oficial de Mozilla Foundation. http://www.mozilla.org/

9. FirebugWorkingGroup. Página web oficial de Firebug. http://getfirebug.com/

10. Wikipedia. Git. http://es.wikipedia.org/wiki/Git

11. Twitter Inc. Página web oficial Twitter Bootstrap. http://twitter.github.com/Bootstrap/